UDACITY

PROJECT REVIEW
CODE REVIEW
NOTES

SHARE YOUR ACCOMPLISHMENT! 🐦 📘

Meets Specifications

Dear student,

You made it! Well done addressing the last remaining issue and completing your project. Aside from posting some pro tips there's not much I can actually further add to your good work though, if you are enthusiastic about process optimization, when it comes to machine learning, I left an advanced Pro Tip for you in the performance section. Please be advised that it is quite advanced, therefore don't worry if you don't get it immediately, you could keep it and use it later on as you progress in the Nanodegree.

Congratulations on passing your exam!

Quality of Code

> **Code reflects the description in the answers to questions in the writeup. i.e. code performs the functions documented in the writeup and the writeup clearly specifies the final analysis strategy.**

> **poi_id.py can be run to export the dataset, list of features and algorithm, so that the final algorithm can be checked easily using tester.py.**

Understanding the Dataset and Question

> **Student response addresses the most important characteristics of the dataset and uses these characteristics to inform their analysis. Important characteristics include:**
>
> - **total number of data points**
> - **allocation across classes (POI/non-POI)**
> - **number of features used**
> - **are there features with many missing values? etc.**

Pro Tip: There are several other options to deal with missing values like:
a. Replacing the values with means or medians.
b. Remove the features that have an exceeding number of missing values.
c. More complex approaches rely on analysing the distribution of missing values: https://en.wikipedia.org/wiki/Missing_data
http://scikit-learn.org/stable/modules/preprocessing.html

> **Student response identifies outlier(s) in the financial data, and explains how they are removed or otherwise handled.**

Excellent section: Great job finding not only the TOTAL outlier but TRAVEL AGENCY IN THE PARK and LOCKHART EUGENE E as well.

Optimize Feature Selection/Engineering

At least one new feature is implemented. Justification for that feature is provided in the written response. The effect of that feature on final algorithm performance is tested or its strength is compared to other features in feature selection. The student is not required to include their new feature in their final feature set.

Univariate or recursive feature selection is deployed, or features are selected by hand (different combinations of features are attempted, and the performance is documented for each one). Features that are selected are reported and the number of features selected is justified. For an algorithm that supports getting the feature importances (e.g. decision tree) or feature scores (e.g. SelectKBest), those are documented as well.

Pro Tip: Please note that you can leverage the power of recursive feature selection to automate the selection process and find a good indication of the number of relevant features, here is an example of how the code might look like:

```python
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import StratifiedKFold
from sklearn.feature_selection import RFECV
lr = LogisticRegression()

rfecv = RFECV(estimator=lr, step=1, cv=StratifiedKFold(labels, 50),
          scoring='precision')
rfecv.fit(features, labels)
print("Optimal number of features : %d" % rfecv.n_features_)
print rfecv.support_
features=features[:,rfecv.support_]
# Plot number of features VS. cross-validation scores
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
```

If algorithm calls for scaled features, feature scaling is deployed.

Pick and Tune an Algorithm

At least two different algorithms are attempted and their performance is compared, with the best performing one used in the final analysis.

Pro Tip (Advanced): Xgboost, one of Kaggle's top algorithms.
In the recent years one algorithm emerged as favourite in the machine learning community, it is actually one of the most used in Kaggle: Xgboost.
Here you can find an informative discussion on why that is the case: https://www.quora.com/Why-is-xgboost-given-so-much-less-attention-than-deep-learning-despite-its-ubiquity-in-winning-Kaggle-solutions
The algorithm is not available sci-kit learn, here is how you can start working with it:
http://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/

Response addresses what it means to perform parameter tuning and why it is important.

At least one important parameter tuned with at least 3 settings investigated systematically, or any of the following are true:

- GridSearchCV used for parameter tuning
- Several parameters tuned
- Parameter tuning incorporated into algorithm selection (i.e. parameters tuned for more than one algorithm, and best algorithm-tune combination selected for final analysis).

Validate and Evaluate

At least two appropriate metrics are used to evaluate algorithm performance (e.g. precision and recall), and the student articulates what those metrics measure in context of the project task.

Response addresses what validation is and why it is important.

Performance of the final algorithm selected is assessed by splitting the data into training and testing sets or through the use of cross validation, noting the specific type of validation performed.

Pro Tip (Advanced): You could actually fully exploit pipelines and parameterize the whole process though cross validation: It means you can try out several algorithms automatically! Watch out though this is pretty advanced stuff, here is a great, informative, top notch tutorial from Zac Sewart!
http://zacstewart.com/2014/08/05/pipelines-of-featureunions-of-pipelines.html

When tester.py is used to evaluate performance, precision and recall are both at least 0.3.

Rate this review